Chapter 3: Introduction to SQL

Database System Concepts, 7th Ed.

Nested Subqueries

- SQL provides a mechanism for the nesting of subqueries. A **subquery** is a **select-from-where** expression that is nested within another query.
- The nesting can be done in the following SQL query

  select A_1, A_2, ..., A_n
  from r_1, r_2, ..., r_m
  where P

  as follows:

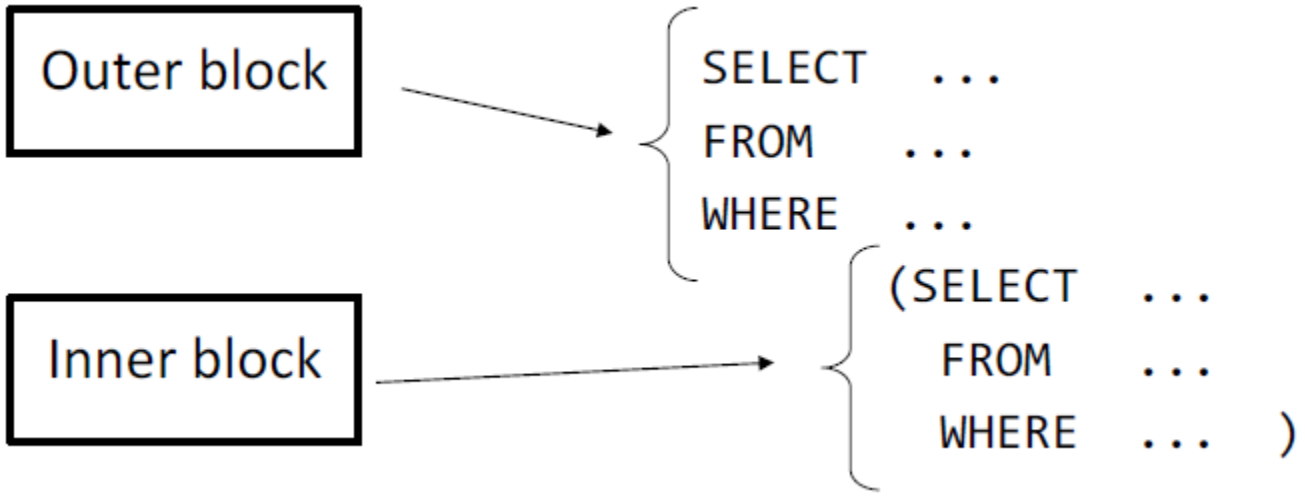
  - **From clause:** r_i can be replaced by any valid subquery
  - **Where clause:** P can be replaced with an expression of the form:

    B <operation> (subquery)

    B is an attribute and <operation> to be defined later.
  - **Select clause:**

    A_i can be replaced be a subquery that generates a single value.

Nested Subqueries



Subqueries in the Select Clause

Scalar Subquery

- Scalar subquery is one which is used where **a single value is expected**
- List all departments along with the number of instructors in each department

```
select dept_name,
        ( select count(*)
           from instructor
           where department.dept_name = instructor.dept_name)
         as num_instructors
from department;
```

- Runtime error if subquery returns more than one result tuple

Scalar Subquery

SID	FIRST	LAST	EMAIL	GRP	MONITOR
101	Ann	Smith	a@jav.co.c	1	(null)
103	Richard	Turner	b@jav.co.c	2	101
102	Michael	Jones	(null)	2	103
104	Ann	Brown	ca@jav.co.c	3	103

- Obtenga las definitivas del curso de cada uno de los estudiantes. Liste el nombre y apellido del estudiante y la sumatoria de puntos de todos los ejercicios (la definitiva del curso). En el resultado no aparece el sid 104

```
SELECT
    s.first,
    s.last,
    SUM(r.points) as definitiva
FROM
    results    r,
    students   s
WHERE
    s.sid = r.sid
GROUP BY
    s.first,
    s.last;
```

FIRST	LAST	DEFINITIVA
Michael	Jones	28
Ann	Smith	30
Richard	Turner	12

```
SELECT
    s.first, s.last,
    (
        SELECT
            SUM(points)
        FROM
            results r
        WHERE
            r.sid = s.sid
    ) AS definitiva
FROM
    students   s
```

FIRST	LAST	DEFINITIVA
Ann	Smith	30
Michael	Jones	28
Richard	Turner	12
Ann	Brown	(null)

```
SELECT
    s.first, s.last,
    nvl((
        SELECT
            SUM(points)
        FROM
            results r
        WHERE
            r.sid = s.sid
    ),0) AS definitiva
FROM
    students   s
```

Subqueries in the From Clause

Subqueries in the Form Clause

- SQL allows a subquery expression to be used in the **from** clause
- Find the average instructors' salaries of those departments where the average salary is greater than $42,000."

```
select dept_name, avg_salary
from ( select dept_name, avg (salary) as avg_salary
          from instructor
          group by dept_name)
where avg_salary > 42000;
```

- Note that we do not need to use the **having** clause
- Another way to write above query

```
select dept_name, avg_salary
from ( select dept_name, avg (salary)
          from instructor
          group by dept_name)
          as dept_avg (dept_name, avg_salary)
where avg_salary > 42000;
```

With Clause

- The **with** clause provides a way of defining a temporary relation whose definition is available only to the query in which the **with** clause occurs.
- Find all departments with the maximum budget

```
with max_budget (value) as
        (select max(budget)
         from department)
select department.name
from department, max_budget
where department.budget = max_budget.value;
```

Complex Queries using With Clause

- Find all departments where the total salary is greater than the average of the total salary at all departments

```
with dept _total (dept_name, value) as
        (select dept_name, sum(salary)
         from instructor
         group by dept_name),
dept_total_avg(value) as
       (select avg(value)
       from dept_total)
select dept_name
from dept_total, dept_total_avg
where dept_total.value > dept_total_avg.value;
```

Ejemplo Esquema Results

```
-- Liste el nombre del estudiante que tiene el mayor numero de puntos obtenidos
--1. calcular los puntos de cada estudiante (group by)
--2. en el query 1 claculo el mayor
--3. obtengo los estudiantes con el puntaje mayor del paso 2
--punto 1
with puntos_por_estudiante as
(
   SELECT s.sid,SUM(r.points) as definitiva
   FROM results r, students   s
   WHERE s.sid = r.sid GROUP BY
s.sid
),
--punto 2
puntaje_mayor as(
    select max(definitiva) as mayor from
  puntos_por_estudiante
)
--select * from puntaje_mayor;
--punto 3
select st.first, st.last, pe.definitiva
from students st,
puntos_por_estudiante pe
where st.sid = pe.sid
and pe.definitiva = (select mayor from
puntaje_mayor)
```

Ejemplo de Agregados con varias columnas

Liste el total de salarios por job y categoria

```
select job,category ,sum(salary) as total
from employees
group by job, category
order by 1,2;
```

esultado de la Consulta

SQL | Todas las Filas Recuperadas: 7 en 0,009 segundos

	JOB	CATEGORY	TOTAL
1	ANALYST	P	3000
2	ANALYST	T	5000
3	CLERK	P	2250
4	MANAGER	T	5300
5	PRESIDENT	P	5000
6	SALESMAN	P	2850
7	SALESMAN	T	1250

Ejemplo de CASE y SUM

Liste el total de salarios por job y categoría, pero de la siguiente manera:

JOB	PLANTA	TEMPORAL
SALESMAN	2850	1250
ANALYST	3000	5000
CLERK	2250	0
MANAGER	0	5300
PRESIDENT	5000	0

```
select job,
sum(case when category = 'P' then salary else 0 end) as Planta,
sum(case when category = 'T' then salary else 0 end) as Temporal
from employees
group by job;
```

Ejemplo de CASE , SUM y totalice

Liste el total de salarios por job y categoría, ahora con fila de TOTALES:

JOB	PLANTA	TEMPORAL
SALESMAN	2850	1250
ANALYST	3000	5000
CLERK	2250	0
MANAGER	0	5300
PRESIDENT	5000	0
Total	24650	24650

```
select nvl(job,'Total') as job,
sum(case when category = 'P' then salary else 0 end) as Planta,
sum(case when category = 'T' then salary else 0 end) as Temporal
from employees
group by ROLLUP(job)
```

Subconsulta Query Externo

```
select job, planta, temporal, planta  + temporal as total
from
(
    select nvl(job,'Total') as job,
    sum(case when category = 'P' then salary else 0 end) as Planta,
    sum(case when category = 'T' then salary else 0 end) as Temporal
    from employees
    group by rollup(job)
)
```

JOB	PLANTA	TEMPORAL	TOTAL
ANALYST	3000	5000	8000
CLERK	2250	0	2250
MANAGER	0	5300	5300
PRESIDENT	5000	0	5000
SALESMAN	2850	1250	4100
Total	13100	11550	24650